HYCON 紘康科技

電壓電流計應用說明書

HY16F198B

Voltage Current Meter

目錄

注意：

1、 本說明書中的內容，隨著產品的改進，有可能不經過預告而更改。請客戶及時到本公司網站下載更新 http://www.hycontek.com。

2、 本規格書中的圖形、應用電路等，因第三方工業所有權引發的問題，本公司不承擔其責任。

3、 本產品在單獨應用的情況下，本公司保證它的性能、典型應用和功能符合說明書中的條件。當使用在客戶的產品或設備中，以上條件我們不作保證，建議客戶做充分的評估和測試。

4、 請注意輸入電壓、輸出電壓、負載電流的使用條件，使 IC 內的功耗不超過封裝的容許功耗。對於客戶在超出說明書中規定額定值使用產品，即使是瞬間的使用，由此所造成的損失，本公司不承擔任何責任。

5、 本產品雖內置防靜電保護電路，但請不要施加超過保護電路性能的過大靜電。

6、 本規格書中的產品，未經書面許可，不可使用在要求高可靠性的電路中。例如健康醫療器械、防災器械、車輛器械、車載器械及航空器械等對人體產生影響的器械或裝置，不得作為其部件使用。

7、 本公司一直致力於提高產品的品質和可靠度，但所有的半導體產品都有一定的失效概率，這些失效概率可能會導致一些人身事故、火災事故等。當設計產品時，請充分留意冗餘設計並採用安全指標，這樣可以避免事故的發生。

8、 本規格書中內容，未經本公司許可，嚴禁用於其他目的之轉載或複製。

1. 簡介

工業上的應用對於電壓及電流的量測，是最基本卻也是最重要的。工業上的壓力、溫度、濕度等.許多測量都是透過感測器後將物理訊號變成電壓或者電流，再透過電子儀器的解析後顯示於儀表上，因此如何量測到精準的電壓、電流是相當重要的。本文主要是介紹 HYCON HY16F198B Series 晶片在電壓電流量測的應用.

由於HY16F198B晶片內部集成高精度ΣΔADC，且ADC輸出頻率最快可以到達10KHz，並搭配內部硬體 LCD 驅動,完成 HY16F198B 用於電壓電流的量測時，擁有相當高的精準度.

2. 原理說明

2.1. 量測原理:

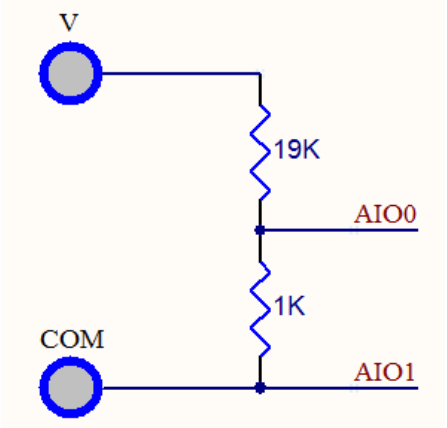


圖1.電壓量測電路圖

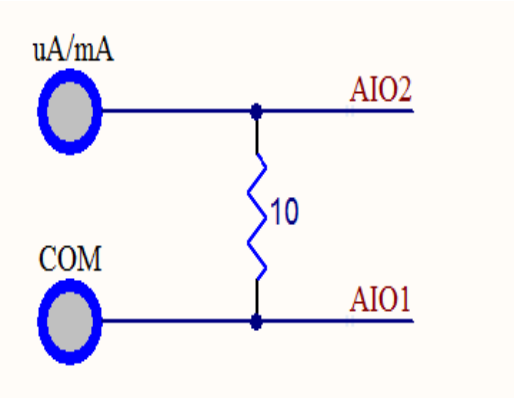


圖2.電流量測電路

2.1.1. 電壓量測:

電路圖如圖1所示，此電路為簡易分壓電路，分壓比例20:1，並由於程式設定關係，AIO0、AIO1 兩端電壓差最大為 1.2V。因此量測電壓上限為 20V.

2.1.2. 電流量測：

電路圖如圖 2 所示，分法為當電流源流過 10Ω 電阻時，產生電壓差。透過量測電壓差方式反推流經電流大小.

解析度分為外部解析度和內部解析度，外部解析度為最大量測的輸出電壓值與需要識別的最小電壓值的電壓值之比，本應用最小量測電壓值為 10mV。

一般我們以目視法認定的內部解析度通常是指我們經軟體處理後LCD顯示只有1格滾動時，此時滿量程的格數就是內部解析度，其 1 格所代表的訊號約為 2-3 倍 RMS Noise。

內外解析度之比越小，電壓電流表精度越高，但內外解析度之比是有限制的。比如滿量程壓差為 1.1V，要做到 2000 Count，內外比為 1：10 的電壓電流表，如果不經過信號放大，

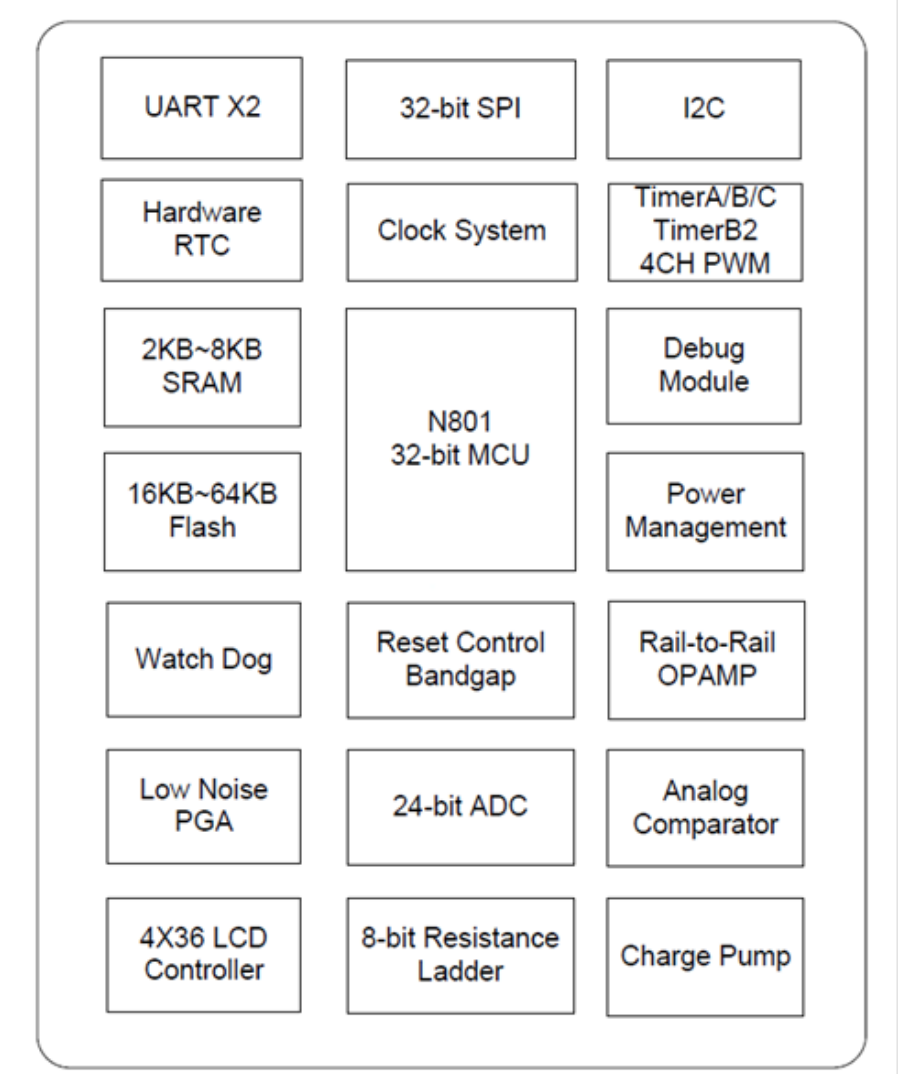
那最小要處理的信號為 1.1V/(2000×10)=55uV。而 SD24 所能處理的最小信號值大約為 65nV，所以要完成此規格的量測示相當容易且精準的。

ADC 性能能否達到規格要求，通常是以 RMS Noise 來推算外部是否穩定內部解析度比值。對於開發電子產品而言，使用 HY16F198B 晶片其所能達到的最大內部解析度的瓶頸在於 Input RMS Noise 而不在於 ADC 的解析度。HY16F198B 的 ADC 待測信號在由 PGA、AD 倍率調整器的放大後（PGA=32，ADGN=4），經 OSR=32768 每秒輸出 10 筆 ADC 值的條件下，其 Input RMS Noise 約為 65nV，但由於其 Input Noise 主要由 Thermal Noise 組成，所以如果我們透過平均的軟體處理是可以再將 Input Noise 進一步降低.

如果我們使用 8 筆的軟體平均處理其 Input RMS Noise 考慮其他雜訊因素後，可達約為 40nV，3 倍 RMS Noise 代表約 1 格的滾動，即為 120nV。在使用 2.4V 驅動電壓，1mV/V 的滿量程時壓差可達 2.4mV，所以在此情形下我們可以得到 20000 Counts 的內部解析度.

2.2. 控制晶片

單片機簡介：HY16F 系列 32 位元高性能 Flash 單片機(HY16F198B)



HY16F 系列 32 位元高性能 Flash 單片機(HY16F198B)

特點說明:

(1)採用最新 Andes 32 位元 CPU 核心 N801 處理器。
(2)電壓操作範圍 2.2~3.6V，以及-40℃~85℃工作溫度範圍。
(3)支援外部 16MHz 石英震盪器或內部 16MHz 高精度 RC 震盪器.
(3.1)運行模式 0.6mA@2MHz/2
(3.2)待機模式 5uA@ LSRC=34KHz+IDLE Mode
(3.3)休眠模式 2.5uA
(4)程式記憶體 64KB Flash ROM
(5)資料記憶體 8KB SRAM
(6)擁有 BOR and WDT 功能，可防止 CPU 死機。
(7)24-bit 高精準度ΣΔADC 類比數位轉換器
(7.1)內置 PGA (Programmable Gain Amplifier)最高可達 128 倍放大。
(7.2)內置溫度感測器 TPS。
(8)超低輸入雜訊運算放大器 OPAMP。
(9)16-bit Timer A
(10)16-bit Timer B 模組俱 PWM 波形產生功能
(11)16-bit Timer C 模組俱數位 Capture/Compare 功能
(12)硬體串列通訊 SPI 模組
(13)硬體串列通訊 I2C 模組
(14)硬體串列通訊 UART 模組
(15)硬體 RTC 時鐘功能模組
(16)硬體 Touch KEY 功能模組
(17)硬體 LCD Driver 4x36,6x34

3. 系統設計

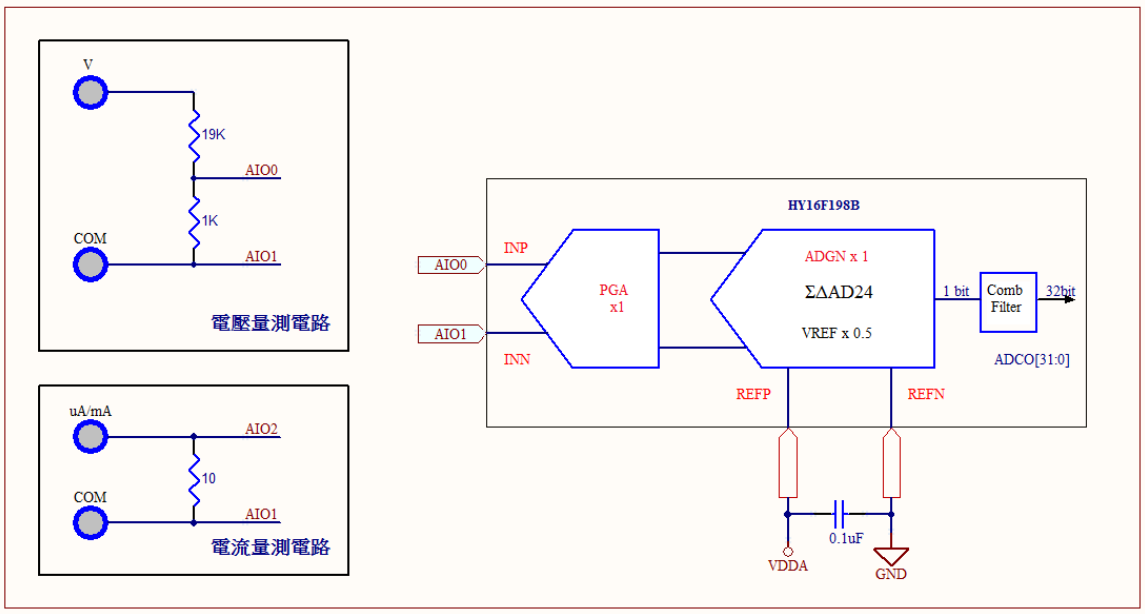
3.1. 硬體說明

使用 HY16F198B 內建 ADC 搭配外部電路進行電壓及電流量測，整體電路包含兩按鈕，分別是(模式選擇)、(測量)按鈕部分，搭配內部硬體 LCD Driver 顯示量測數值.

(A) MCU：HY16F198B
(B) 顯示方式： HY16F198B 內部硬體驅動 4x36 LCD (LCD Driver Segment 4X36)
(C) 電源電路：5.0V 轉 3.3V 電源系統
(D) 類比感測模組：內部 ADC
(E) 線上燒錄與 ICE 連結電路，透過 EDM 的連接，可支援線上燒錄模擬.
並擁有強大的 C 平台 IDE 以及 HYCON 類比軟體分析工具與 GUI 等支援.

3.2. 功能說明

ADC 內部的 PGA 放大 1 倍，ADGN 放大 1 倍，參考電壓由 VDDA –VSS 供給，則 ΔVR_I=1.2V。

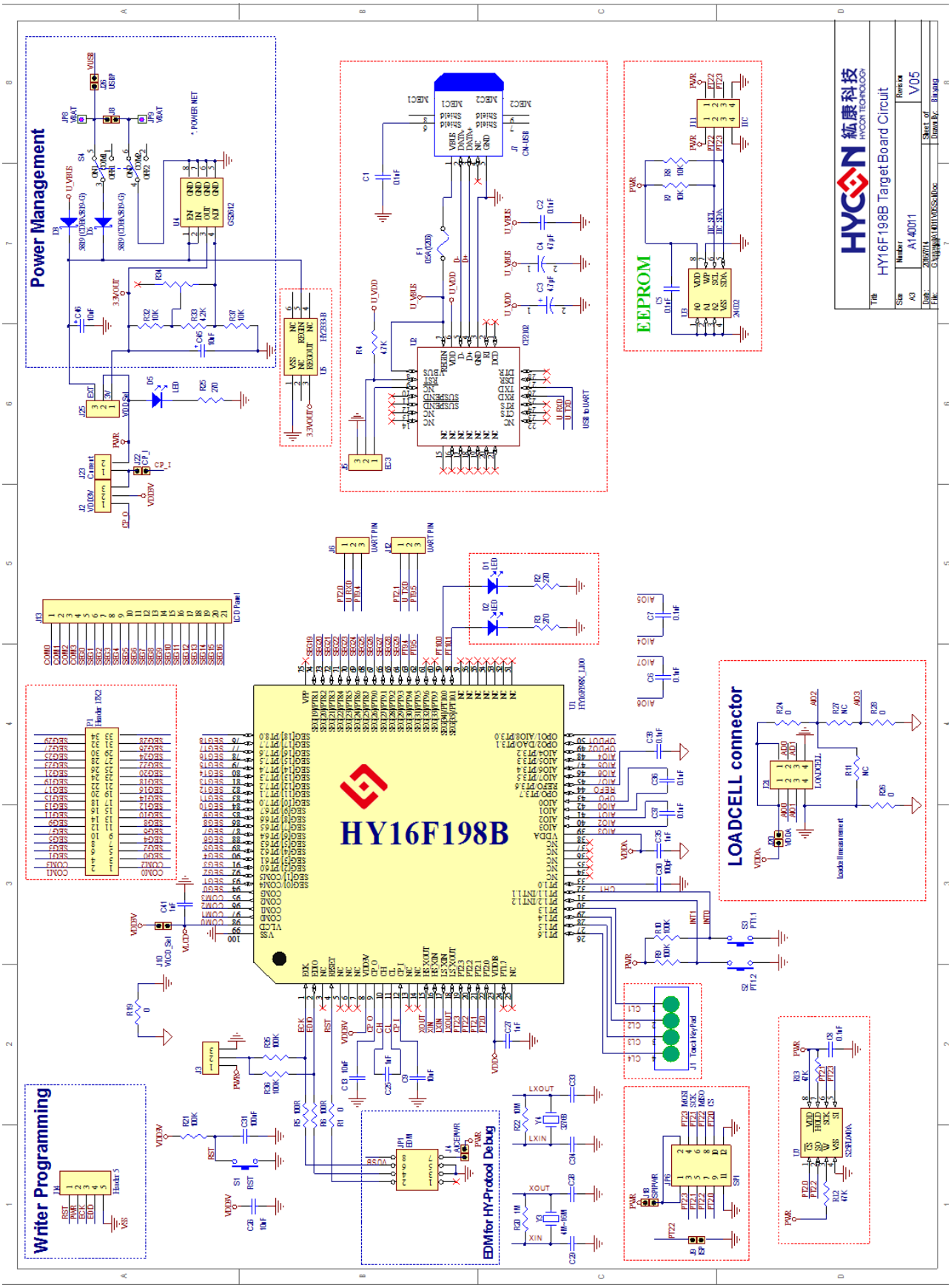


3.2.1. 電壓量測

電壓量測模式下，量測範圍為±20V，搭配電壓量測電路。顯示至 1mV，精準度至 10mV.

3.2.2. 電流量測

電流量測主要範圍為±110mA，搭配電流量測電路。顯示及量測精準度皆為 0.1mA


Power Management
EEPROM
LOADCELL connector
HY16F198B
Writer Programming
EDM for HY-Protool Debug
HY16F198B Target Board Circuit
V05

4. 操作流程

4.1. 操作方法

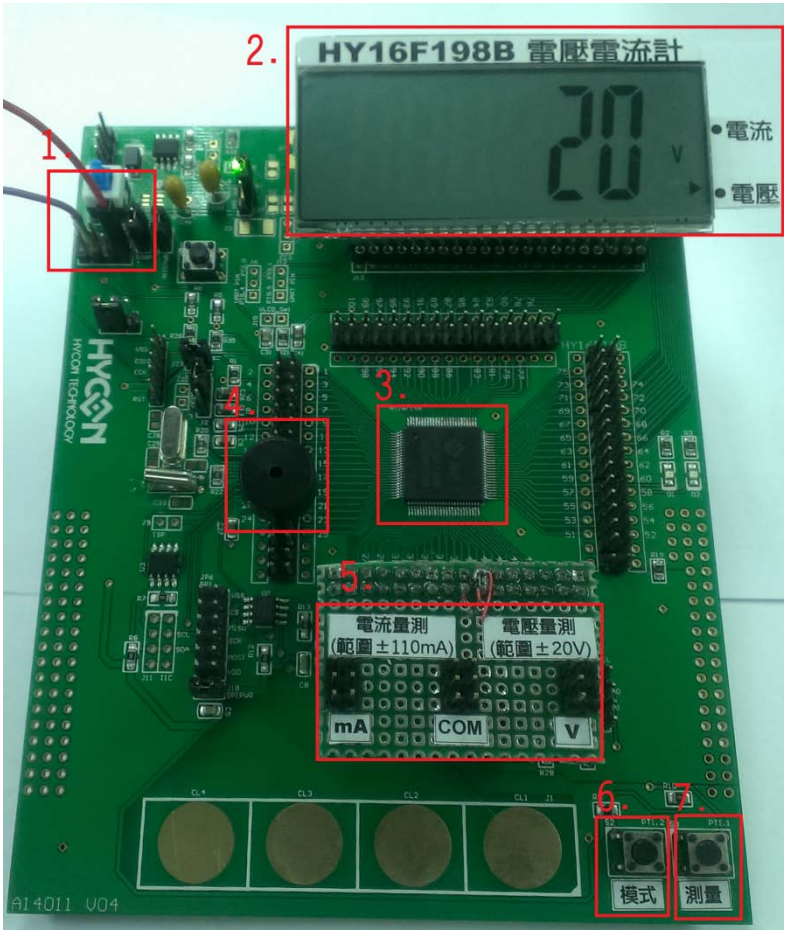
啓動後，首先將 LCD 全點亮，再進行初始化及 Hycon 字樣顯示.之後跳至模式選擇.

4.1.1. 按鍵控制說明

透過 S2(模式按鈕)進行量測模式切換；S3(量測按鈕)代表開始量測.
並且每次按鍵 Buzzer 都會發出聲音.

4.1.2. 測量電壓模式

20V 即代表±20V 量測(搭配外部量測電路)

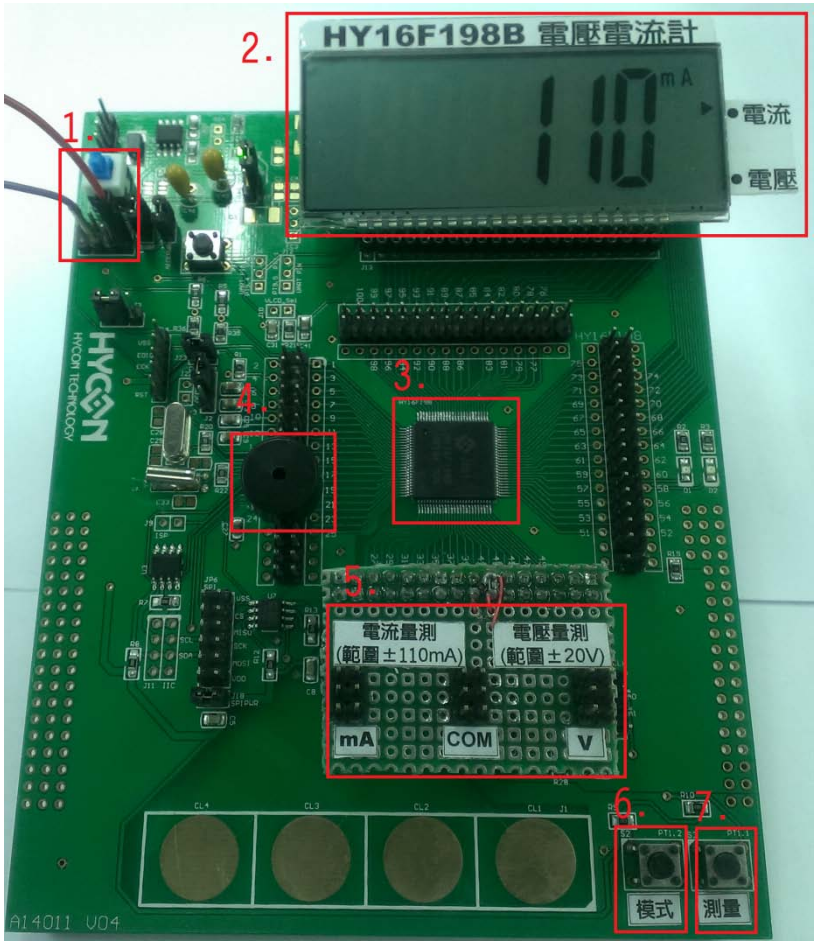


圖片說明:

1. 5V 電源輸入端
2. LCD 顯示電壓模式或量測數值.
3. MCU 位置(HY16F198B)
4. Buzzer 輸出
5. 電壓量測 V & COM (量測範圍+/-20V)
6. 模式選擇 Key
7. 測量 Key

4.1.3. 測量電流模式

110mA 即代表±110mA 量測(搭配外部量測電路)

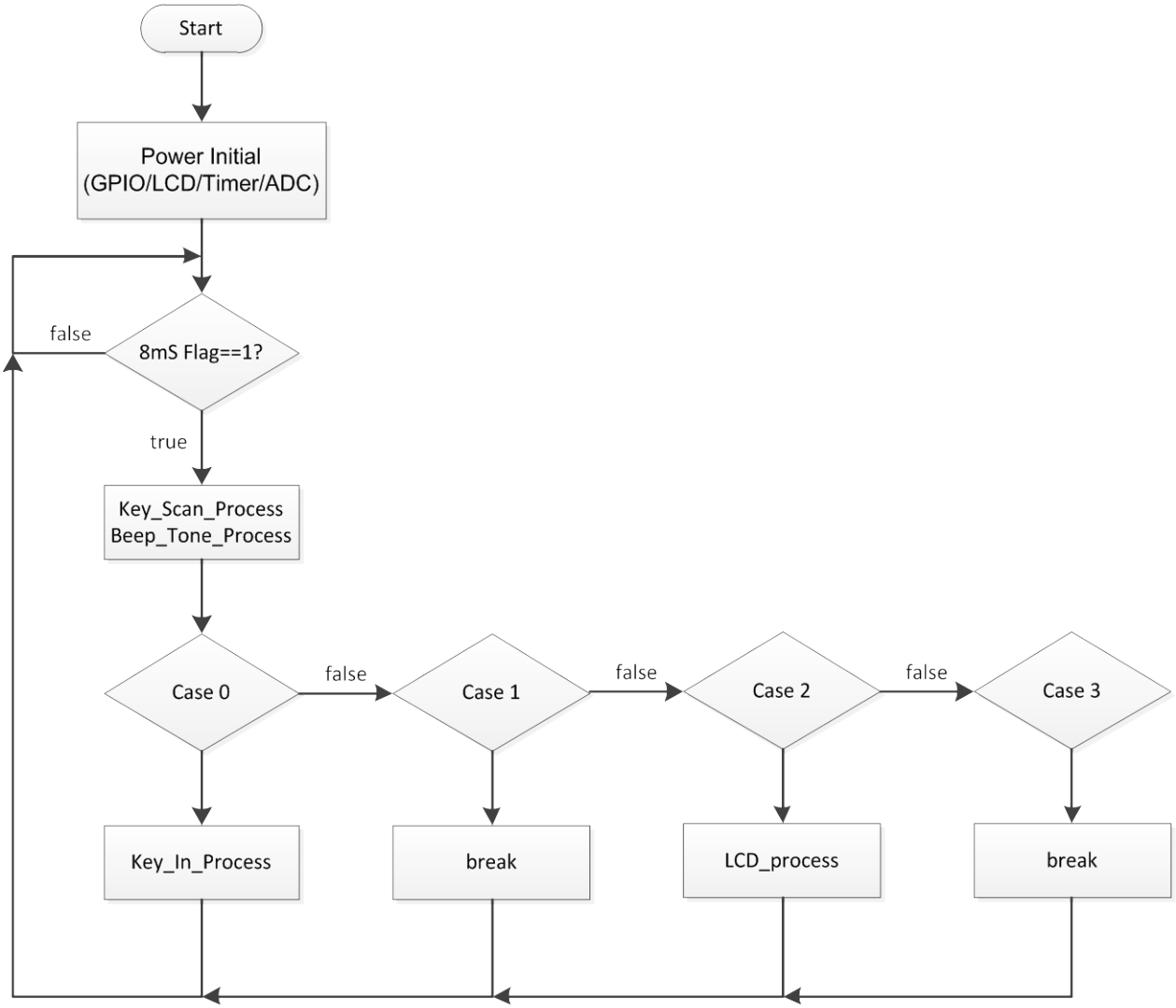


圖片說明:

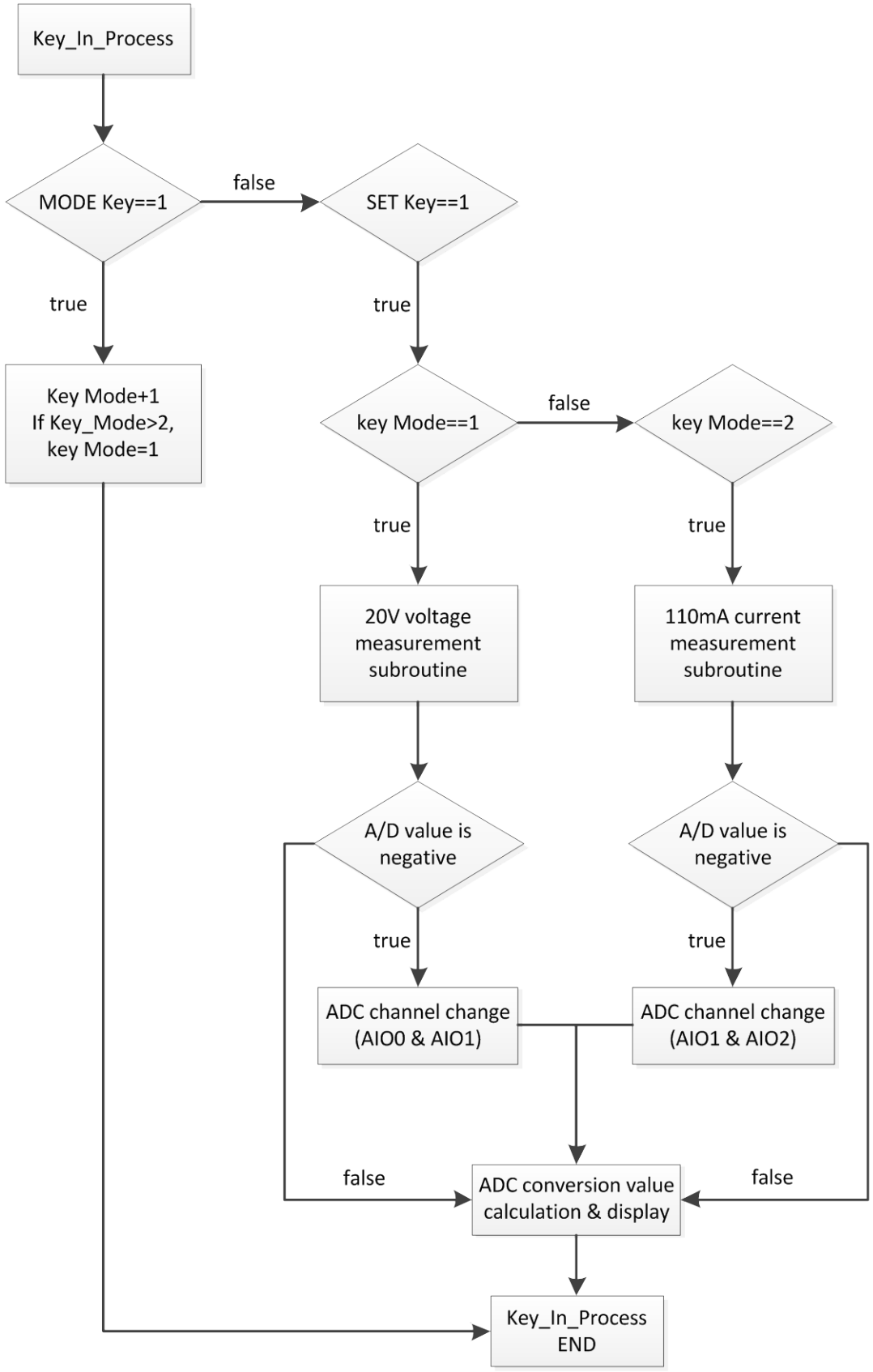
1. 5V 電源輸入端.
2. LCD 顯示為電流模式或量測數值.
3. MCU 位置(HY16F198B)
4. Buzzer 輸出
5. 電流量測 mA & COM (量測範圍+/-110mA)
6. 模式選擇 Key
7. 測量 Key

4.2. 程式流程

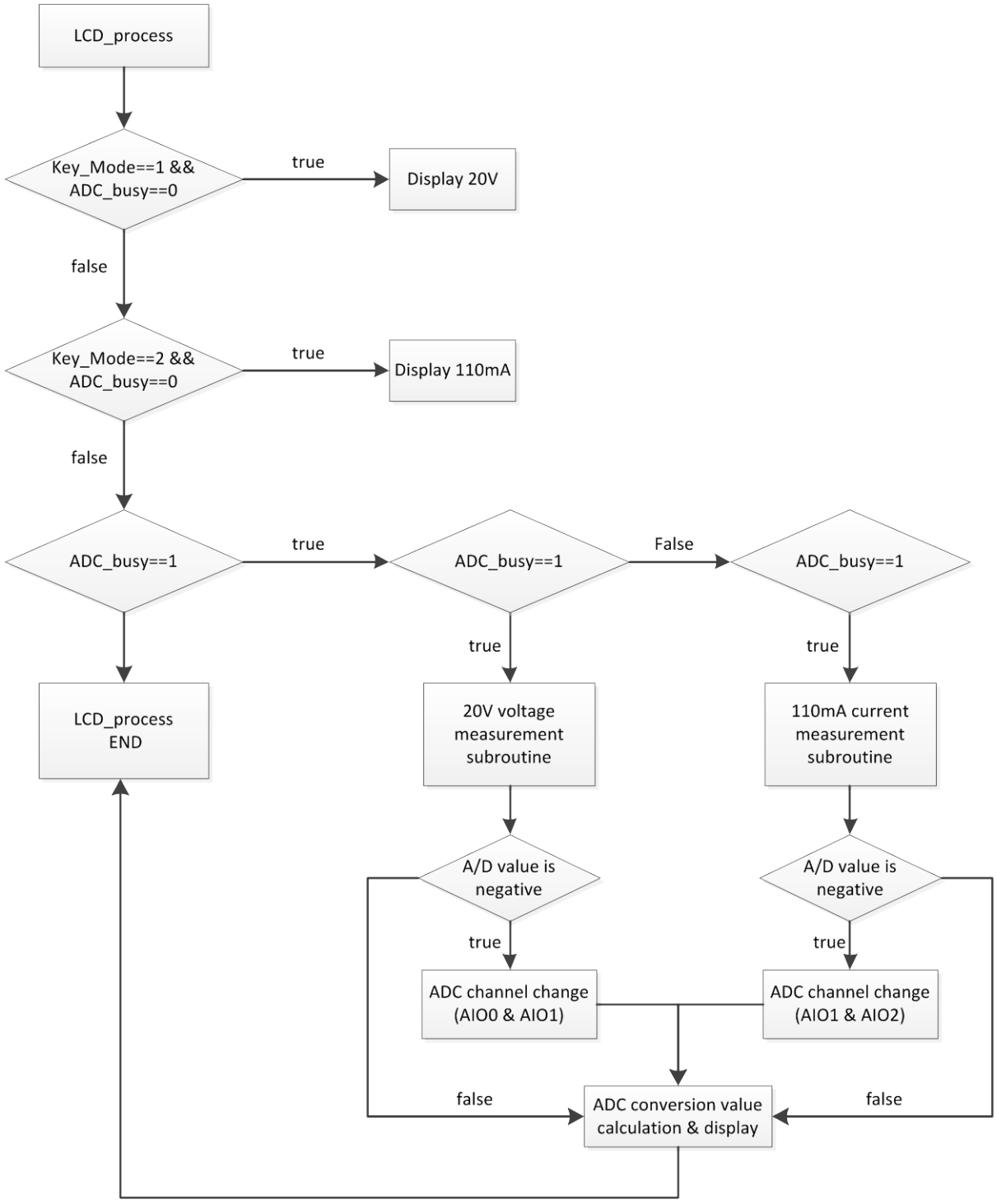
4.2.1. Main Loop 流程圖:



4.2.2. 按鍵處理流程圖:


Key_In_Process
MODE Key==1
false
SET Key==1
true
true
Key Mode+1
If Key_Mode>2,
key Mode=1
key Mode==1
false
key Mode==2
true
true
20V voltage
measurement
subroutine
110mA current
measurement
subroutine
A/D value is
negative
A/D value is
negative
true
true
ADC channel change
(AIO0 & AIO1)
ADC channel change
(AIO1 & AIO2)
false
ADC conversion value
calculation & display
false
Key_In_Process
END

4.2.3. LCD 顯示處理流程圖:



5. 技術規格

(1) VDD=3.3V
(2) 功耗:工作模式約 2.24mA(HAO=4MHz,ADC Enable)
(3) 量測精準度:電壓 10(mV) 以及電流 0.1(mA)
(4) 適用範圍：量測電壓範圍(± 20V)
量測電流範圍(± 110mA)
(5) 工作溫度:-40℃~ +85℃

5.1. 測試數據

20V 電壓測試

Fluke輸出電壓	16F198B測得電壓		誤差%	
mV	+	-	+	-
10	10	10	0.0000	0.0000
20	20	20	0.0000	0.0000
30	30	30	0.0000	0.0000
40	40	40	0.0000	0.0000
50	50	50	0.0000	0.0000
60	60	60	0.0000	0.0000
70	70	70	0.0000	0.0000
80	80	80	0.0000	0.0000
90	90	90	0.0000	0.0000
100	100	100	0.0000	0.0000
200	200	200	0.0000	0.0000
300	300	300	0.0000	0.0000
400	400	400	0.0000	0.0000
500	500	500	0.0000	0.0000
600	600	600	0.0000	0.0000
700	700	700	0.0000	0.0000
800	800	800	0.0000	0.0000
900	900	900	0.0000	0.0000
1000	1000	1000	0.0000	0.0000
1500	1500	1500	0.0000	0.0000
1600	1600	1600	0.0000	0.0000
1700	1700	1700	0.0000	0.0000
1800	1800	1800	0.0000	0.0000
1900	1900	1900	0.0000	0.0000

2000	2000	2000	0.0000	0.0000
2500	2500	2500	0.0000	0.0000
3000	3000	3000	0.0000	0.0000
3500	3500	3500	0.0000	0.0000
4000	4000	4000	0.0000	0.0000
4500	4500	4500	0.0000	0.0000
5000	5000	5000	0.0000	0.0000
6000	6000	6000	0.0000	0.0000
7000	7000	7000	0.0000	0.0000
8000	8000	8000	0.0000	0.0000
9000	9000	9000	0.0000	0.0000
10000	10000	10000	0.0000	0.0000
11000	11000	11000	0.0000	0.0000
12000	12000	12000	0.0000	0.0000
13000	13000	13000	0.0000	0.0000
14000	14000	14000	0.0000	0.0000
15000	15000	15000	0.0000	0.0000
16000	16000	16000	0.0000	0.0000
17000	17000	17000	0.0000	0.0000
18000	18000	18000	0.0000	0.0000
19000	19000	19000	0.0000	0.0000
20000	20000	20000	0.0000	0.0000

110mA 電流測試

Fluke輸出電流	16F198B測得電流		誤差%	
mA	+	-	+	-
0	0	0	0	0
1	1	1	0	0
2	2	2	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
6	6	6	0	0
7	7	7	0	0
8	8	8	0	0
9	9	9	0	0
10	10	10	0	0
11	11	11	0	0
12	12	12	0	0
13	13	13	0	0
14	14	14	0	0
15	15	15	0	0
16	16	16	0	0
17	17	17	0	0
18	18	18	0	0
19	19	19	0	0
20	20	20	0	0
25	25	25	0	0
30	30	30	0	0
40	40	40	0	0
50	50	50	0	0
60	60	60	0	0
70	70	70	0	0
80	80	80	0	0
90	90	90.1	0	0.111
100	100	100.1	0	0.1
110	110.1	110.1	0.09	0.09

6. 結果總結

以 HY16F198B 為主控結合內部高精度、多通道輸入、快速 ADC 的量測。不論電壓或者電流的量測，相較於市售電表，不僅僅耗電量低於一般市售電表，在精準度上也有不輸市售電表的表現。HY16F198B 內部 ADC 不僅可用來量測電壓電流，也可以結合外部感測器進行其他量測，依然有相當不錯的表現.

7. 附件

7.1. 範例程式

```
/*---------------------------------------------------------------------------*/
// Header file include
/*---------------------------------------------------------------------------*/
#include "HY16F198.h"
#include "System.h"
#include "ModuleID.h"
#include "SpecialMacro.h"
#include "Sysinfra.h"
#include "DrvClock.h"
#include "DrvGPIO.h"
#include "DrvLCD.h"
#include "DrvREG32.h"
#include "DrvADC.h"
#include "DrvTimer.h"
#include "DrvUART.h"
#include "user_DEF.h"
#include "DrvPMU.h"
/*---------------------------------------------------------------------------*/
// STRUCTURES
/*---------------------------------------------------------------------------*/
#if (1)
volatile typedef union _MCUSTATUS
{
     char  _byte;
     struct
     {
          unsigned b_ADCdone:1;
          unsigned b_TMAdone:1;
          unsigned b_TMBdone:1;
          unsigned b_TMC0done:1;
          unsigned b_TMC1done:1;
          unsigned b_RTCdone:1;
          unsigned b_UART_TxDone:1;
          unsigned b_UART_RxDone:1;
     };
} MCUSTATUS;
#endif
/*---------------------------------------------------------------------------*/
// Global CONSTANTS
/*---------------------------------------------------------------------------*/
MCUSTATUS  MCUSTATUSbits;

extern unsigned char seg[16];

/*---------------------------------------------------------------------------*/
// Constant value definition
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
// Function prototype declaration
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
// MAIN  function
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
// Function PROTOTYPES
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
// MAIN  function
/*---------------------------------------------------------------------------*/
int main(void)
{
     unsigned char scan_idx = 0;

     User_Sample_Init();

     //Enter Main program loop
     while(1)
     {
```

```
          //User task initial going ?
          if( UserFlag0.b_InitGo_Flag == TRUE )
               PowerOn_Init_Go();
          //each task processing period= (base time) * 4tasks = 8mS * 4 = 32mS.

          if( UserFlag0.b_TB8mS_Flag == TRUE )
          {
               UserFlag0.b_TB8mS_Flag = FALSE;
               Key_Scan_Process();                    //key matrix scanning.
               Beep_Tone_Process();                   //Beep tone process !

               scan_idx = (scan_idx+1) & 0x03;        //keep bit[1:0].
               switch( scan_idx )
               {
                    //Key matrix scanning.
               case 0:
                    Key_In_Process();                 //Key in processing.
                    break;
               case 1:
                    break;
                    //LCD display !
               case 2:
                    LCD_DATA_DISPLAY2();
                    break;
                    //Sleep Time Update counting.
               case 3:
               {
                    if ( 0 == Sleep_Time_Cnt )
                         UserFlag0.b_EntSLP_Flag = TRUE;
                    else
                         Sleep_Time_Cnt--;
                    break;
               }
               default:
                    break;
               }
          }
          //drv_wdt_clr();                           //Clear WDT Timer !
     }
     return 0;
}

/*-------------------------------------------------------------------------*/
// Exception Service Routines
/*-------------------------------------------------------------------------*/
void tlb_exception_handler()
{
     //procedure define by customer.
     asm("nop");
     asm("nop");
}


/*************************************************************************/
// File Name    : user_ISR.c
// Description  : This file implements the customer's ISR function.
/*************************************************************************/
/*-------------------------------------------------------------------------*/
// Function Name: HW1_ISR()
// Description  : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).
/*-------------------------------------------------------------------------*/
void HW1_ISR(void)
{
     if( DrvTIMER_GetIntFlag( E_TMA ) )
     {
          DrvTIMER_ClearIntFlag( E_TMA );        //Clear TMA interrupt flag
          UserFlag0.b_TB8mS_Flag = TRUE;
          //pio2_2 ^= 0x0303;
     }
}
/*-------------------------------------------------------------------------*/
// Function Name: HW2_ISR()
// Description  : ADC interrupt Service Routine (HW2).
/*-------------------------------------------------------------------------*/
void HW2_ISR(void)
{
     DrvADC_ClearIntFlag();
     ADCData=adc_08;
```

```
    ok=1;
}

/*****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Structure definition                                                      */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Global variable definition                                                */
/*---------------------------------------------------------------------------*/

unsigned int  Sleep_Time_Cnt;
unsigned char DisplayBuffer[18];

unsigned char Key_DB_Cnt;
unsigned int  NoKey_Reset_Cnt;
unsigned int  NoKeyBuf[4];
unsigned char NoBuf_Idx;

unsigned char Beep_Tone_Cnt;
unsigned char Key_Code_Buf;
unsigned char Key_Code_No;

unsigned int  ADC_busy;
unsigned char i;

int ADCData;
#define  Disable 0
#define  Enable  1
unsigned int key_Mode;
signed int sum,ADtemp10,ADtemp101,ADtemp100;      //AD用
signed int ADtemp20n,ADtemp20p,ADtemp200,ADtemp110n,ADtemp110p,ADtemp1100,average;    //計算及顯示用
int nu,ch,ok;                                                 //判斷用
int value_buf[16];

/*---------------------------------------------------------------------------*/
/* Constant value definition                                                 */
/*---------------------------------------------------------------------------*/
unsigned char seg[]=
{
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f,         // char "0"
    seg_b+seg_c,                                 // char "1"
    seg_a+seg_b+seg_d+seg_e+seg_g,               // char "2"
    seg_a+seg_b+seg_c+seg_d+seg_g,               // char "3"
    seg_b+seg_c+seg_f+seg_g,                     // char "4"
    seg_a+seg_c+seg_d+seg_f+seg_g,               // char "5"
    seg_a+seg_c+seg_d+seg_e+seg_f+seg_g,         // char "6"
    seg_a+seg_b+seg_c,                           // char "7"
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f+seg_g,  // char "8"
    seg_a+seg_b+seg_c+seg_d+seg_f+seg_g,         // char "9"
    seg_a+seg_b+seg_c+seg_e+seg_f+seg_g,         // char "A"
    seg_c+seg_d+seg_e+seg_f+seg_g,               // char "b"
    seg_a+seg_d+seg_e+seg_f,                     // char "C"
    seg_b+seg_c+seg_d+seg_e+seg_g,               // char "d"
    seg_a+seg_d+seg_e+seg_f+seg_g,               // char "E"
    seg_a+seg_e+seg_f+seg_g                      // char "F"
};

/*---------------------------------------------------------------------------*/
/* Function prototype declaration                                            */
/*---------------------------------------------------------------------------*/

//------MISC Function declaration---------------
void User_Sample_Init(void);
void Delay(unsigned int num);
void PowerOn_Init_Go(void);

void Beep_Tone_Start(void);
void Beep_Tone_Stop(void);
void Beep_Tone_Process(void);

//------ADC Function declaration----------------
void InitalADC(void);
//------GPIO Function declaration---------------
void Key_Scan_Process(void);
void Key_In_Process(void);
```

```
void GPIO_Init(void);

//------TIMER Function declaration-------------
void TIMER_Init(void);
//------LCD Function declaration---------------
void LCD_Init(void);
void LCD_DATA_DISPLAY(unsigned int LcdBuffer);
void RAM2LCD(unsigned char *Buffer_Adr );
void ClearLCDframe(void);
void ALLLCDframe(void);
void DisplayHYcon(void);
void LCD_DATA_DISPLAY1(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY2(void);

void AD_average(void);                          //AD平均副程式
void AD_symbol_voltage(void);                   //偵測電壓正負使用
void AD_symbol_curreunt(void);                  //偵測電流正負使用
void Voltage_measurement(void);                 //電壓量測副程式
void Current_measurement(void);                 //電流量測副程式



/*---------------------------------------------------------------------------*/
// Function Name: User_Sample_Init
/*---------------------------------------------------------------------------*/
void User_Sample_Init(void)
{
     SYS_DisableGIE();        //Disable Global Interrupt.

//------Sys. clock setting----------------------
     DrvCLOCK_SelectIHOSC( TRIM_HAO4MHZ );   //Select HAO= 4MHz.
     DrvCLOCK_EnableHighOSC(E_INTERNAL, 60); //Enable HAO.
     DrvCLOCK_SelectMCUClock(0, 0);               //Select MCUCKS= HS_CK/1.

//------MISC. system initial--------------------
     GPIO_Init();
     LCD_Init();
     TIMER_Init();
     UserFlag0._byte = 0;
//------Power on initial going------------------
//Beep tone on & LCD display HYcon SCALE about 1Sec.
     ALLLCDframe();
     Delay(0x40000);
     DisplayHYcon();
     Delay(0x40000);
     UserFlag0.b_InitGo_Flag = TRUE;
     SYS_EnableGIE( 4, 0x06 );                   //Enable HW1 & HW2 Int. only.
}

/*---------------------------------------------------------------------------*/
// Function Name: PowerOn_Init_Go
// Description  : Initial all user tasks (likes LCD, ADC for scale etc),
/*---------------------------------------------------------------------------*/
void PowerOn_Init_Go(void)
{
     unsigned int i;
     InitalADC();
     //enable Comb Filter.
     DrvADC_CombFilter( 0 );              //reset !
     DrvADC_CombFilter( 1 );              //enable !
     for(i=0; i<16; i++)
     {
          value_buf[i] = 0;
     }
     ADtemp200=0;
     ADtemp1100=0;
     ADtemp110n=1405;
     ADtemp110p=1405;
     ADtemp20n=7;
     ADtemp20p=7;
     nu=0;
     key_Mode=1;
     ch=0;
     ADC_busy=0;
     UserFlag1._byte = 0;
     UserFlag0.b_InitGo_Flag = FALSE;
```

```
}

/*---------------------------------------------------------------------------*/
// Function Name: TIMER_Init
// Description  : TIMER Initialization Subroutines.
/*---------------------------------------------------------------------------*/
void TIMER_Init(void)
{
//------Initial TIMER A for Time Base-----------
     DrvTMA_Open( 9,0 );                                              //Set TimerA int
period= HS_CK/32/1024= 4MHz/32768= 8mS.
     DrvTIMER_ClearIntFlag( E_TMA );                                  //Clear Timer A
interrupt flag
     DrvTIMER_EnableInt( E_TMA );                                   //Timer A interrupt enable

//------Initial TIMER B for PWM-----------------
     DrvTMBC_Clk_Source( E_HS_CK, 0 );                              //set TMB Clock from
HS_CK/1.
     DrvTMB_Open( E_TMB_MODE0, E_TMB_NORMAL, PWM_4KHZ_TIME );         //set TMB working on normal mode
& TMB clock int period= PWM_4KHZ_TIME/HS_CK= 250uS.
}


/*---------------------------------------------------------------------------*/
// Function Name: Delay()
// Description  : Software delay subroutines.
// Remark       : num= 10,   delay time= 18u S @4MH HAO, 3V.
//              : num= 100,  delay time= 40u S @4MH HAO, 3V.
//              : num= 1000, delay time= 260uS @4MH HAO, 3V.
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
     for( ; num >0; num-- )
          asm( "NOP" );
}

/*****************************************************************************/
// File Name    : user_GPIO.c
// Description  : This file implements the customer's GPIO function.
/*****************************************************************************/

/*---------------------------------------------------------------------------*/
// Function Name: InitalADC
// Description  : ADC Initialization Subroutines
/*---------------------------------------------------------------------------*/

void InitalADC(void)
{
     //Set ADC input pin
     DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);//ADC Input
     DrvADC_InputSwitch(OPEN);
     DrvADC_RefInputShort(OPEN);
     DrvADC_Gain(0,0);                          //Set the ADC Gain
     DrvADC_DCoffset(0);                        //DC offset input voltage selection
     DrvADC_RefVoltage(VDDA,0);                 //Set the ADC reference voltage.
     DrvADC_FullRefRange(1);                         //Set the ADC reference range select.
                                                          //0:Full reference range input
                                                          //1:1/2 reference range input
     DrvADC_OSR(1);                                  //1:OSR=16384
     DrvADC_ClkEnable(1,1);                          //set ADC CLOCK ADCK = HS_CK/12 = 4MHz/12 = 333KHz
& rising edge is high.
     DrvPMU_VDDA_Voltage(E_VDDA2_4);
     DrvPMU_VDDA_LDO_Ctrl(E_LDO);
     DrvPMU_BandgapEnable();
     DrvPMU_REFO_Enable();
     DrvPMU_AnalogGround(Enable);               //ADC analog ground source selection.
                                                          //1:Enable buffer and use internal source
     DrvPMU_LDO_LowPower(0);                    //VDD LDO with low power control.
                                                          //0 : Normal;1 : Low power

     Delay(0x1000);
     DrvADC_EnableInt();
     DrvADC_ClearIntFlag();
     DrvADC_Enable();
}

/*---------------------------------------------------------------------------*/
// Function Name: LCD_Init
/*---------------------------------------------------------------------------*/
```

```
void LCD_Init(void)
{

     clk_10 = 0x40404B4B;                    //Set LCK= (HS_CK/64)/9/16= (HAO/64)/9/16= (4MHz/64)/144=
434Hz & charge pump clock = HS_CK/16.
     DrvLCD_VLCDMode( E_VLCD27 );
     DrvLCD_VLCDTrim( 3 );                    //Trim VLCD as 2.93V.
     DrvLCD_LcdDuty ( E_LCD_DUTY4 );           //enable LCD@1/4 duty.
     DrvLCD_IOMode( 5, 0xFF );                //Set SEG1/0 as LCD mode.
     DrvLCD_IOMode( 0, 0xFF );                //Set PT6(SEG2 ~SEG9 ) as LCD mode.
     DrvLCD_IOMode( 1, 0x7F );                //Set PT7(SEG10~SEG16) as LCD mode.
     ClearLCDframe();
     DrvLCD_LCDBuffer( ENABLE );              //enable LCD working.
     DrvLCD_DisplayMode( E_LCD_NORMAL );       //LCD at normal mode.
}

/*---------------------------------------------------------------------------*/
// Clear LCD RAM Data
/*---------------------------------------------------------------------------*/
void ClearLCDframe(void)
{
     int i=0;
     for(i=0; i<18; i++)
     {
          DisplayBuffer[i]=0x00;
     }
     RAM2LCD( DisplayBuffer );
}


/*---------------------------------------------------------------------------*/
// SET LCD RAM Data
/*---------------------------------------------------------------------------*/
void ALLLCDframe(void)
{
     int i=0;
     for(i=0; i<18; i++)
     {
          DisplayBuffer[i]=0xFF;
     }
     RAM2LCD( DisplayBuffer );
}


/*---------------------------------------------------------------------------*/
// Display HYcon Char
/*---------------------------------------------------------------------------*/
void DisplayHYcon(void)
{
     DisplayBuffer[0]=0x00;
     DisplayBuffer[1]=Char_H;
     DisplayBuffer[2]=Char_Y;
     DisplayBuffer[3]=Char_c;
     DisplayBuffer[4]=Char_o;
     DisplayBuffer[5]=Char_n;
     DisplayBuffer[6]=0x00;
     DisplayBuffer[7]=0x00;
     DisplayBuffer[8]=0x00;
     RAM2LCD( DisplayBuffer );
}

/*---------------------------------------------------------------------------*/
// Function Name: RAM2LCD
/*---------------------------------------------------------------------------*/
void RAM2LCD( unsigned char *Buffer_Adr )
{
     unsigned char buf_idx;

     for(buf_idx=0; buf_idx<SEG_BUF_SIZE; buf_idx++)
     {
          DrvLCD_WriteData ( buf_idx, *Buffer_Adr );
          Buffer_Adr++;
     }
}

/*---------------------------------------------------------------------------*/
// Function Name: Key_Scan_Process
/*---------------------------------------------------------------------------*/
```

```
void Key_Scan_Process(void)
{
     unsigned char key_code, temp;
     unsigned int  r_bit;
     key_code = NULL_KEY_NO;
     temp = NULL_KEY_NO;

     if ( !(r_bit=DrvGPIO_GetBit(E_PT1,2)))       // Check S2 (MODE)
     {
          key_code = 0x00;
     }
     if ( !(r_bit=DrvGPIO_GetBit(E_PT1,1)))       // Check S3  (SET)
     {
          key_code = 0x01;
     }

//------Key in debounce-------------------------
     Key_DB_Cnt++;
     if ( key_code != Key_Code_Buf )
     {
          Key_Code_Buf = key_code;
          Key_DB_Cnt = 0;
     }
     else
     {
          if (Key_DB_Cnt >= KEY_DB_TIME)
          {
               Key_DB_Cnt = 0;
               if ( Key_Code_Buf == NULL_KEY_NO )
               {
                    UserFlag0.b_KeyOn_Flag = FALSE;
               }
               else
               {
                    if (UserFlag0.b_KeyOn_Flag == FALSE)
                    {
                         UserFlag0.b_KeyOn_Flag = TRUE;
                         UserFlag0.b_KInPro_Flag = TRUE;
                         Key_Code_No = Key_Code_Buf;
                    }
                    Sleep_Time_Cnt = SLEEP_TIME;
               }
          }
     }
     DrvGPIO_PT1_DisableOUTPUT( 0xFF );      //close PT1[7:0] OE.
}

/*---------------------------------------------------------------------------*/
// Function Name: GPIO_Init
/*---------------------------------------------------------------------------*/
void GPIO_Init(void)
{
    DrvGPIO_ClkGenerator( E_HS_CK, 3 );                    //Set IO sampling clock input source is HS_CK,
and IOCLK is HS_CK/4.
//  DrvGPIO_ClkGenerator( E_HS_CK, 11);                    //Set IO sampling clock input source is HS_CK,
and IOCLK is HS_CK/1024.
     DrvGPIO_PT1_IntTriggerPorts( 0x0F, E_N_Edge );           //PT1[3:0] interrupt trigger method is
negative edge.
     DrvGPIO_PT1_DisableINT( 0xFF );                        //PT1[7:0] interrupt disable.
     DrvGPIO_PT1_ClearIntFlag( 0xFF );                      //clear PT1[7:0] interrupt flag.
     DrvGPIO_PT1_EnablePullHigh( 0x06 );                    //enable PT1[3:0] pull high 75K ohm.(set
PT1.1 PT1.2)
     DrvGPIO_PT1_EnableINPUT( 0x06 );                       //set PT1[3:0] as input port. (set PT1.1
PT1.2)
     DrvGPIO_PT1_DisableOUTPUT( 0xFF );                     //close Output PT1[7:0] OE.

//------PWM function initial--------------------
     DrvGPIO_Open( E_PT2, 0x03, E_IO_OUTPUT );              //set PT2[1:0] as output port.
     DrvPWM_CountCondition( PWM_4KHZ_TIME/2, PWM_4KHZ_TIME/2 );  //Set TBC1 as 50% duty, TBC2 as 50% duty.
     DrvGPIO_ClrPortBits( E_PT2, 0x03 );                    //set PT2[1:0] as output low.
}



/*---------------------------------------------------------------------------*/
// Function Name: Beep_Tone_Start
/*---------------------------------------------------------------------------*/
void Beep_Tone_Start(void)
```

```
{
     Beep_Tone_Cnt = BEEP_TONE_TIME;
     DrvTMB_Open( E_TMB_MODE0, E_TMB_NORMAL, PWM_4KHZ_TIME );  //set TMB working on normal mode & TMB
clock int period= PWM_4KHZ_TIME/HS_CK= 250uS.
     DrvPWM0_Open( 0, 1, 2 );                                  //Set PT2.0= PWMO0, PT2.1= PWMO1 and PWMO0
working on PWMA mdoe with normal pulse output.
     DrvPWM1_Open( 0, 0, 2 );                                  //Set PT2.0= PWMO0, PT2.1= PWMO1 and PWMO1
working on PWMA mdoe with reverse pulse output.
}

/*---------------------------------------------------------------------------*/
// Function Name: Beep_Tone_Stop
/*---------------------------------------------------------------------------*/
void Beep_Tone_Stop(void)
{
     Beep_Tone_Cnt = 0;
     DrvTMB_Close();
     DrvPWM0_Close();
     DrvPWM1_Close();
     DrvGPIO_ClrPortBits( E_PT2, 0x03 );       //set PT2[1:0] as output low.
}

/*---------------------------------------------------------------------------*/
// Function Name: Beep_Tone_Process
/*---------------------------------------------------------------------------*/
void Beep_Tone_Process(void)
{
     if (Beep_Tone_Cnt != 0)
     {
          Beep_Tone_Cnt--;
          if (Beep_Tone_Cnt <= 0)
               Beep_Tone_Stop();
     }
}

/*---------------------------------------------------------------------------*/
// Function Name: Key_In_Process                                             */
/*---------------------------------------------------------------------------*/
void Key_In_Process(void)
{
     unsigned int i;
//------Rest for newly No. key in---------------
     NoKey_Reset_Cnt++;
     if ( NoKey_Reset_Cnt > NO_KEY_RESET_TIME )
     {
          UserFlag1.b_DotKey_Flag = FALSE;
          NoKey_Reset_Cnt = 0;
          NoBuf_Idx = 2;
          NoKeyBuf[0] = 0;
          NoKeyBuf[1] = 0;
          NoKeyBuf[2] = 0;
          NoKeyBuf[3] = 0;
     }
     if( UserFlag0.b_KInPro_Flag != TRUE )
          return;
     UserFlag0.b_KInPro_Flag = FALSE;

     switch( Key_Code_No )
     {
     case 0:                                   //Mode select key
          ADC_busy=0;
          key_Mode++;
          if(key_Mode>=3)     key_Mode=1;      //Mode 1: voltage check
          asm("nop");                          //Mode 2: current check
          nu=0;
          break;

     case 1:                                   //Set key
     {
          switch(key_Mode)
          {
          case 0x00:
               break;
          case 0x01:
               DrvADC_Disable();
               DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);
               DrvADC_Enable();
               DrvADC_CombFilter( 0 );       //reset !
```

```
                DrvADC_CombFilter( 1 );       //enable !
                for(i=0; i<16; i++)
                {
                     value_buf[i] = 0;
                }
                Delay(0x1000);
                Voltage_measurement();
                ADC_busy=1;
                break;

           case 0x02:
                DrvADC_Disable();
                DrvADC_SetADCInputChannel(ADC_Input_AIO2,ADC_Input_AIO1);
                DrvADC_Enable();
                DrvADC_CombFilter( 0 );       //reset !
                DrvADC_CombFilter( 1 );       //enable !
                for(i=0; i<16; i++)
                {
                     value_buf[i] = 0;
                }
                Delay(0x1000);
                Current_measurement();
                ADC_busy=1;
                break;

           default:
                break;
           }
      }
      default:
           asm("nop");
           break;
      }
      Beep_Tone_Start();
}

/*---------------------------------------------------------------------------*/
// Function Name: Voltage_measurement   (20V電壓量測副程式)
/*---------------------------------------------------------------------------*/
void Voltage_measurement(void)
{
      AD_symbol_voltage();                          //呼叫正負電壓判定副程式
      if(nu==0)                                     //輸入為正顛壓
      {
           AD_average();                            //呼叫AD副程式
           if(sum<131072)
           {
                ADtemp101=100*(sum-ADtemp200)/(ADtemp20n-ADtemp200);
                ADtemp100=ADtemp101%100;
                if(ADtemp100>70) ADtemp101+=100;
                ADtemp101=ADtemp101/100;
                ADtemp101=ADtemp101*10;
                ADtemp101=ADtemp101>>3;
                ADtemp101=ADtemp101/10;
                ADtemp101=ADtemp101*10;
                if(ADtemp101<9)
                {
                     ADtemp101=0;
                     nu=0;
                }
                LCD_DATA_DISPLAY(ADtemp101);          //電壓顯示
           }
           else
           {
                if(sum<131172)
                {
                     ADtemp101=100*(sum-ADtemp200)/(ADtemp20n-ADtemp200);
                     ADtemp101*=10;
                     ADtemp101=ADtemp101>>3;
                     ADtemp101=ADtemp101/10;
                     ADtemp101=ADtemp101*10;
                }
                else
                {
                     nu=0;
                     ADtemp101=0;
                }
```

```
                LCD_DATA_DISPLAY(ADtemp101);
            }
        }
        if(nu==1)                                           //輸出為負電壓
        {
            AD_average();                                   //呼叫AD副程式
            if(sum<56080)
            {
                ADtemp101=100*(sum+ADtemp200)/(ADtemp20n+ADtemp200);
                ADtemp100=ADtemp101%10;
                if(ADtemp100>70) ADtemp101+=100;
                ADtemp101=ADtemp101/100;
                ADtemp101=ADtemp101*10;
                ADtemp101=ADtemp101>>3;
                ADtemp101=ADtemp101/10;
                ADtemp101=ADtemp101*10;
                if(ADtemp101<10)
                {
                    ADtemp101=0;
                    nu=0;
                }
                LCD_DATA_DISPLAY(ADtemp101);            //電壓顯示
            }
            else
            {
                if(sum<114050)
                {
                    ADtemp101=(sum+ADtemp200)/(ADtemp20n+ADtemp200);
                    ADtemp101*=10;
                    ADtemp101=ADtemp101>>3;
                    ADtemp101=ADtemp101/10;
                    ADtemp101=ADtemp101*10;
                }
                else
                {
                    nu=0;
                    ADtemp101=0;
                }
                LCD_DATA_DISPLAY(ADtemp101);
            }
        }
}


/*-------------------------------------------------------------------------------*/
// Function Name: Current_measurement   (110mA電流量測副程式)
/*-------------------------------------------------------------------------------*/
void Current_measurement(void)
{
    AD_symbol_curreunt();
    if(nu==0)                                               //輸入為正顛壓
    {
        AD_average();
        ADtemp101=100*(sum+1)/(ADtemp110n-ADtemp1100);      //數值運算
        ADtemp101=ADtemp101>>3;
        LCD_DATA_DISPLAY1(ADtemp101);                       //current display
    }
    if(nu==1)
    {
        AD_average();
        ADtemp101=100*(sum+1)/(ADtemp110n-ADtemp1100);
        ADtemp101=ADtemp101>>3;
        if(ADtemp101==0)
        {
            nu=0;
        }
        LCD_DATA_DISPLAY1(ADtemp101);
    }
}

/*-------------------------------------------------------------------------------*/
// Function Name: AD_average  (ADC副程式)
/*-------------------------------------------------------------------------------*/
void AD_average(void)
{
    unsigned char t;
    while(ok)                                               //ADC中斷後OK=1
```

```
    {
        ok=0;
        ADtemp10=(ADCData>>14);
        value_buf[11]=value_buf[10];
        value_buf[10]=value_buf[9];
        value_buf[9]=value_buf[8];
        value_buf[8]=value_buf[7];                    //平均滑動濾波處理
        value_buf[7]=value_buf[6];
        value_buf[6]=value_buf[5];
        value_buf[5]=value_buf[4];
        value_buf[4]=value_buf[3];
        value_buf[3]=value_buf[2];
        value_buf[2]=value_buf[1];
        value_buf[1]=ADtemp10;
        sum=0;
        for(t=8; t>0; t--)
        {
            sum+=value_buf[t];
        }
        average=((sum)>>3);
    }
    if(average<0)
    {
        average*=-1;
    }
    sum=average;
}

/*---------------------------------------------------------------------------*/
// Function Name: AD_symbol_voltage     (電壓正負號判定副程式)
/*---------------------------------------------------------------------------*/
void AD_symbol_voltage(void)
{
    int t;
    sum=0;
    for(t=0; t<8; t++)                                          //將AD輸入值8筆取平均
    {
        ADtemp10=(ADCData>>17);
        sum+=ADtemp10;
    }
    if(sum>=-30)                                                //現況輸入判定
    {
        if(ch==0)                                               //前一次判定為正
        {
            nu=0;                                               //目前輸入為正
            ch=0;
            DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1);   //AIO0、AIO1正常配置
            goto by;                                            //離開此程式
        }
        if(ch==1)                                               //前一次判定為負
        {
            nu=1;                                               //目前輸入為負
            ch=1;
            DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);   //AIO0、AIO1顛倒配置
            goto by;
        }
    }
    if(sum<0)                                                   //現況輸入判定
    {
        if(ch==0)                                               //前一次輸入為正
        {
            nu=1;                                               //現況判定為負
            ch=1;
            DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);   //AIO0、AIO1顛倒配置
            goto by;
        }
        if(ch==1)                                               //前一次輸注為負
        {
            nu=0;                                               //現況判定為正
            ch=0;
            DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1);   //AIO0、AIO1正常配置
            goto by;
        }
    }
by:
```

```
     asm("NOP");
     Delay(0x1000);
}

/*---------------------------------------------------------------------------*/
// Function Name: AD_symbol_curreunt    (電流正負號判定副程式)
/*---------------------------------------------------------------------------*/
void AD_symbol_curreunt(void)
{
     int t;
     sum=0;
     for(t=0; t<8; t++)                                          //將AD輸入值8筆取平均
     {
          ADtemp10=(ADCData>>17);
          sum+=ADtemp10;
     }
     if(sum>=-30)                                                //現況輸入判定
     {
          if(ch==0)                                              //前一次判定為正
          {
               nu=0;                                             //目前輸入為正
               ch=0;
               DrvADC_SetADCInputChannel(ADC_Input_AIO2,ADC_Input_AIO1);    //AIO2、AIO1正常配置
               goto by1;                                         //離開此程式
          }
          if(ch==1)                                              //前一次判定為負
          {
               nu=1;                                             //目前輸入為負
               ch=1;
               DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO2);    //AIO2、AIO1顛倒配置
               goto by1;
          }
     }
     if(sum<0)                                                   //現況輸入判定
     {
          if(ch==0)                                              //前一次輸入為正
          {
               nu=1;                                             //現況判定為負
               ch=1;
               DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO2);    //AIO2、AIO1顛倒配置
               goto by1;
          }
          if(ch==1)                                              /前一次輸注為負
          {
               nu=0;                                             //現況判定為正
               ch=0;
               DrvADC_SetADCInputChannel(ADC_Input_AIO2,ADC_Input_AIO1);    //AIO2、AIO1正常配置
               goto by1;
          }
     }
by1:
     asm("NOP");
     Delay(0x1000);
}

/*---------------------------------------------------------------------------*/
// Function Name: LCD_DATA_DISPLAY (顯示電壓量測數值)
/*---------------------------------------------------------------------------*/
void LCD_DATA_DISPLAY(unsigned int LcdBuffer)          //顯示選單
{
     DisplayBuffer[0]=0x00;
     DisplayBuffer[1]=0x00;
     DisplayBuffer[2]=0x00;
     DisplayBuffer[3]=0x00;
     DisplayBuffer[4]=0x00;
     DisplayBuffer[5]=0x00;
     int i;
     for(i=5; i>=0; i--)
     {
          DisplayBuffer[i]=seg[LcdBuffer%10];
          LcdBuffer=LcdBuffer/10;
     }

     if(nu==1) DisplayBuffer[6]=S_Minus;               //負號
     else DisplayBuffer[6]=0x00;
```

```
     DisplayBuffer[7]=S_V;                                  //電壓的V
     DisplayBuffer[2]=DisplayBuffer[2]|seg_h;               //小數點
     DisplayBuffer[8]=S21;
     RAM2LCD( DisplayBuffer );
}
/*---------------------------------------------------------------------------*/
// Function Name: LCD_DATA_DISPLAY1     (顯示電流量測數值)
/*---------------------------------------------------------------------------*/
void LCD_DATA_DISPLAY1(unsigned int LcdBuffer)
{

     DisplayBuffer[0]=0x00;
     DisplayBuffer[1]=0x00;
     DisplayBuffer[2]=0x00;
     DisplayBuffer[3]=0x00;
     DisplayBuffer[4]=0x00;
     DisplayBuffer[5]=0x00;
     int i;
     for(i=5; i>=0; i--)
     {
          DisplayBuffer[i]=seg[LcdBuffer%10];
          LcdBuffer=LcdBuffer/10;
     }
     if(nu==1) DisplayBuffer[6]=S_Minus;                    //負號
     else DisplayBuffer[6]=0x00;
     DisplayBuffer[7]=S_A;                                  //電流的A
     DisplayBuffer[6]=DisplayBuffer[6]|S_m1;                //m
     DisplayBuffer[4]=DisplayBuffer[4]|seg_h;               //小數點
     DisplayBuffer[8]=S19;
     RAM2LCD( DisplayBuffer );
}
/*---------------------------------------------------------------------------*/
// Function Name: LCD_DATA_DISPLAY2     (顯示目前MODE&偵測目前電壓或電流)
/*---------------------------------------------------------------------------*/
void LCD_DATA_DISPLAY2(void)
{
     if(key_Mode==1 && ADC_busy==0)
     {
          DisplayBuffer[0]=0x00;
          DisplayBuffer[1]=0x00;
          DisplayBuffer[2]=0x00;
          DisplayBuffer[3]=0x00;
          DisplayBuffer[4]=seg[2];                          //顯示20V
          DisplayBuffer[5]=seg[0];
          DisplayBuffer[6]=DisplayBuffer[6]&0xF6;
          DisplayBuffer[7]=S_V;
          DisplayBuffer[8]=S21;
     }
     if(key_Mode==2 && ADC_busy==0)
     {
          DisplayBuffer[0]=0x00;
          DisplayBuffer[1]=0x00;
          DisplayBuffer[2]=0x00;
          DisplayBuffer[3]=seg[1];                          //顯示110mA
          DisplayBuffer[4]=seg[1];
          DisplayBuffer[5]=seg[0];
          DisplayBuffer[6]=S_m1;
          DisplayBuffer[7]=S_A;
          DisplayBuffer[8]=S19;
     }
     if(ADC_busy==1)                                            //偵測目前電壓或電流
     {
          switch(key_Mode)
          {
          case 0x01:
               Voltage_measurement();
               break;                                           //呼叫20V電壓量測副程式

          case 0x02:
               Current_measurement();
               break;                                           //呼叫110mA電流量測副程式

          default:
               break;
          }
     }
```

```
    RAM2LCD( DisplayBuffer );
}

/*-------------------------------------------------------------------------*/
/* End Of File                                                             */
/*-------------------------------------------------------------------------*/
```

7.2. 附加檔案

APD-HY16F022_DemoCode_V01.zip

8. 參考文獻

[1] http://www.hycontek.com/attachments/MSP/APD-HY16F008_TC.pdf
紘康科技HY16F188 電壓電流計

[2] http://www.hycontek.com/attachments/MSP/DS-HY16F198_TC.pdf
紘康科技HY16F198 Datasheet.

[3] http://www.hycontek.com/attachments/MSP/UG-HY16F198_TC.pdf
紘康科技 HY16F198 User Guide.

9. 修訂記錄

以下描述本檔差異較大的地方，而標點符號與字形的改變不在此描述範圍。

日期	文件版次	頁次	摘要
2016/06/30	V1.0	ALL	初版發行